Towards the UAL Open Source Project

N. Malitsky, R. Talman, M. Blaskiewicz, R. Calaga, R. Fliller III,
A. Luccio, T. Satogata, J. Wei, BNL, Upton, NY 11973, USA

Abstract

Unified Accelerator Libraries (UAL[1]) software has been introduced as an open accelerator simulation environment providing support for many-to-many associations between diverse accelerator algorithms and diverse accelerator applications. Recently, UAL has been successfully applied to the development and study of the SNS Ring realistic beam dynamics modelincluding a complex combination of several physical effects and dynamic processes (such as injection painting, field errors, space charge effects, impedances, fringe fields, misalignments, etc.). The SNS and previous applications have confirmed the major UAL conceptual solutions and have encouraged us to transform this software into an Open Source project[2]. The major efforts have been releasing documentation and consolidation of UAL modules based on the Accelerator Propagator Framework (APF). At this time, the documentation encompasses User Guide, API specification of C++ classes, Perl User interface, and a collection of feature-illustrating examples. Also APF has been implemented to enhance the UAL infrastructure by providing a uniform mechanism for development and integration of accelerator algorithms. The key part of this approach is the Accelerator Propagator Description Format (APDF) that provides physicists a mechanism for switching among simulation models within their applications.

SNS RING APPLICATION

The need to reduce beam losses to parts per ten thousand in the SNS high intensity proton accelerator complex have introduced a new level of requirements and expectations for beam dynamics studies. Realistic predictions at this level of precision demand a close reproduction of a complex combination of effects and dynamic processes in the accelerator simulation model. To address these tasks, the SNS Ring Accelerator Physics Group developed the SNS Ring package based on the UAL simulation environment[3]. Topics to which the package has been applied include[4][5]:

- optimization of injection painting schemes;
- nonlinear effects arising from kinematics terms, magnet imperfections, and fringe fields;
- dynamic aperture and diffusion map studies;
- effect of space charge during transverse painting;
- tune spreads from space charge, chromaticity, and other nonlinearity in combination;
- intensity limitation and choice of working point dictated by imperfection resonance crossing in the presence of space charge;
- half-integer coherent resonance crossing;
- collective instability due to transverse coupling impedance;
- halo development and beam loss modeling.

These intensive studies required the deployment of the UAL software on parallel clusters. The original architecture was comfortably fitted to the Message-Passing Interface (MPI) parallel environment without any changes of existing modules. Then the time consuming algorithms were implemented as extensions (C++ shared libraries) and combined with other sequential and parallel components.

ACCELERATOR PROPAGATOR FRAMEWORK

The extensibility of the UAL environment is provided by its main architectural principle: separation of propagators from accelerator elements. This approach enables one to apply a variety of different simulation modules to the same accelerator lattice. Having initially rejected any implicit linkage between algorithm and element, the Accelerator Propagator Framework defines a mechanism for connecting accelerator elements with propagators tailored to each particular simulation model. In order to describe the structure of the simulation model, we have introduced the Accelerator Propagator Description Format (APDF). One can consider the APDF file to be a complement to the MAD lattice file. Its structure and relationship to elements and algorithms are indicated in Fig. 1.

Just as the initial lattice description unwinds into a (long) ordered list of all elements in the lattice, the propagator builder associates an appropriate propagator with every element in this list. But, as Fig.1 indicates, default associations permit the APDF file to be quite brief. Some of the possible algorithms are indicated in the figure. “MltTracker” and “DriftTracker” implement pure, element-by-element, kick tracking, for example through elements “qd1” and “sd1”, by virtue of their element type being either “quadrupole” or “sextupole”. “SectorTracker” implements concatenated, matrix or nonlinear mapping, for example from just before element “d1” to just before element “qf1”. Tracking algorithm can also be associated with element based on the element name; for example the “BPM” algorithm is associated with element “bpm1” in Fig.1. This facilitates special processing at particular elements. Like

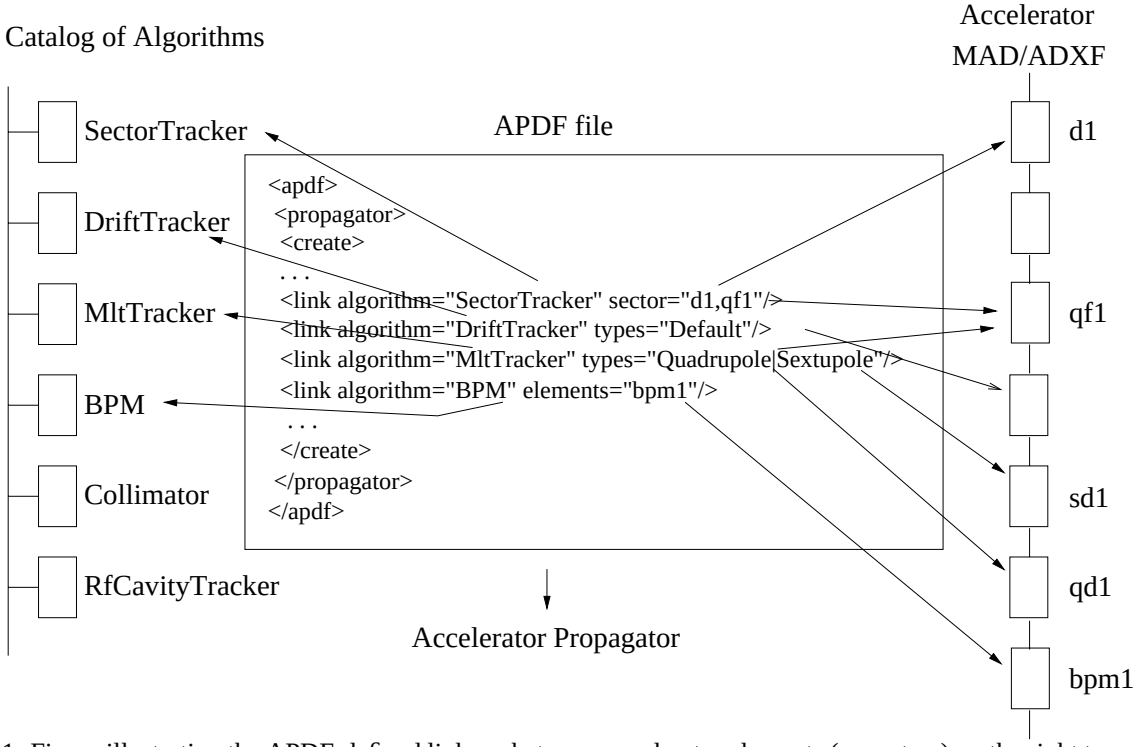

Figure 1: Figure illustrating the APDF-defined linkage between accelerator elements (or sectors) on the right to propagation elements on the left.

the MAD format, APDF addresses a spectrum of applications ranging from small special tasks to full-scale, realistic model encompassing heterogeneous algorithms and special effects. Some possible modeling scenarios are indicated in Table1, which is intended to be self-explanatory. Many of these scenarios have been applied within UAL in the past, but only as dedicated applications. The APDF provides these capabilities without any additional programming complication. Examples have been:

1 Longitudinal beam dynamics	2D matrices + RF tracker
2 Linear lattice functions	4D matrices
3 Fast tracking with chromatic effects (Fast Teapot)	6D matrices with chromatic extensions + selected quad, sext, RF trackers
4 Instumentation modeling, e.g. beam transfer function	#3 + propagators for active diagnostic devices, such as AC dipole
5 Dynamic aperture, halo, IR background investigation	element-type associations
6 Special localized effect, beam-beam, impedance, ions	#3 or #5 + propagator for special effect
7 Near-symplectic large amplitude modeling	octupole imperfection maps, plus kick tracking
8 Full-scale ‘‘realistic’’	all of the above

UAL ARCHITECTURE

The organization of the UAL components is indicated schematically in Fig. 2

At this time, the APF-based modules included in UAL are:

- ZLIB: numerical library for differential algebra[6]

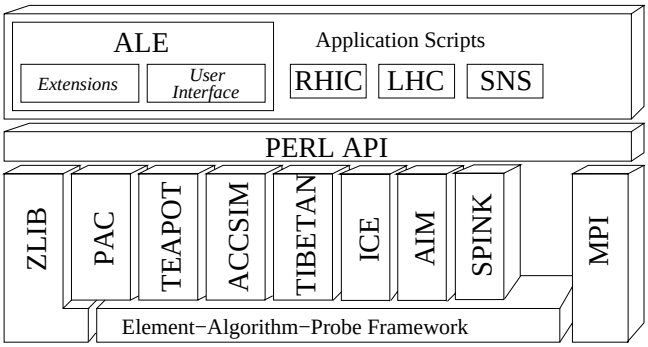

Figure 2: UAL architecture. The figure represents dependency metaphorically, by gravity; codes appearing higher up are supported by (that is, use) codes further down. The upper levels of the figure indicate control via scripting language (PERL).

- PAC: Platform for Accelerator Codes[7]
- TEAPOT: Thin Element Program for Optics and Tracking[8]
- ACCSIM : Accelerator Simulation Code[9]
- TIBETAN : longitudinal phase space tracking program[10]

Modules that are partially supported and are under active development are

- SPINK: tracking code for polarized particles in a circular accelerator[11]

- ICE: Incoherent and Coherent Effects[12]
- AIM: Accelerator Instrumentation Module

The Application Programming Interface (API), written in Perl, provides a universal shell for integrating and managing all project extensions. Consolidation of C++ interfaces has also created a basis for supporting Swig-based interfaces to other script languages (e.g. Python).

POST PROCESSING APPLICATIONS

Another potential benefit of an environment such as UAL is the feasability and economy of "infrastructure" (shared resources) such as postprocessors, plotting/histogramming/fitting for visualization, input and output translation, and parallel processing.

One example of this sort has been preliminary integration of the UAL environment with the ROOT environment[13]. ROOT is an open source project that has been used for many years by high energy nuclear and particle physics experiments for data and simulation analysis. It consists of a C/C++ interpreter CINT and a large number of C++ classes implementing fitting, graphing, GUI, mathematics and various programming functions. A C/C++ interpeter offers an alternative approach to traditional scripting languages, such as PERL or PYTHON, and allows physicists and developers to use a single programming language for an entire project. This environment is especially appropriate for detector background investigations conducted jointly by detector shielding groups and accelerator physicists because ROOT is so well established in the particle physics sector. In one such investigation a proposed RHIC collimator setup was investigated by processing UAL tracking results with the ROOT toolkit. The resulting particle flux distributions can then be passed to experimental physicists for their simulation of detector background.

Another post processing example involved the investigation of Model Independent Analysis (MIA[14]), starting from the following (complete) APDF file:

```
<apdf>
  <propagator>
    <link algorithm="TEAPOT::DriftTracker"
          types= "Default" />
    <link algorithm="TEAPOT::DipoleTracker"
          types="SBend" />
    <link algorithm="TEAPOT::MltTracker"
          types="Quadrupole|Sextupole|Multipole|
                 [VH]kicker|Kicker" />
    <link algorithm="TIBETAN::RfCavityTracker"
          types="RfCavity" />
    <link algorithm="MIA::BPM"
                types="Monitor" />
  </propagator>
</apdf>
```

To simulate MIA, multiturn output from T turns at each of B BPM's was recorded and the resulting TxB matrix was subsequently subjected to singular value decomposition analysis to extract the fundamental modes of the accelerator. When the lattice Twiss functions were reconstructed from the extracted phases at every BPM they were in excellent agreement with the lattice functions determined directly from the original lattice model.

Another post processing approach is to launch a graphing program from within the UAL PERL script. For nonlinear analysis a by-now standard approach is to subject turn-by-turn data to FFT analysis, to extract the tunes by peak location and then to identify accelerator resonances as sum or difference frequencies. The program GRACE[15] makes all these capabilities available and provides graphical output either file driven or via pipe. Zooming, panning, labeling and other prettification and output of the graphs can then be performed, completely independent of UAL, using routine GRACE capabilities. Phase space plots normalized by calculated Twiss parameters can be produced similarly.

REFERENCES

[1] N. Malitsky and R. Talman, *Unified Accelerator Libraries*, AIP 391, 1996.

[2] See `http://www.ual.bnl.gov`. References to original reports and publications are contained in the User Guide accessible from that source: N. Malitsky and R. Talman, UAL User Guide, BNL Formal Report 71010-2003, 2002

[3] N.Malitsky et al, *Application of UAL to High-Intensity Beam Dynamics Studies in the SNS Accumulator Ring*, EPAC 2002.

[4] A.V.Fedotov et al. *Effect of Nonlinearities on Beam Dynamics in the SNS Accumulator Ring*, EPAC00, p. 1492.

[5] A.V.Fedotov et al. *Excitation of Resonances Due to the Space Charge and Magnet Errors in the SNS Ring*, PQC01, p. 2878.

[6] Y. Yan and C-Y. Yan *Numerical Library for Differential Algebra*, SSCL-300, 1990.

[7] N. Malitsky, A. Reshetov, and G. Bourianoff, *Platform for Accelerator Codes*, SSCL-675, 1994.

[8] L. Schachinger and R. Talman, *TEAPOT: A Thin Element Program for Optics and Tracking*, Part. Accel. **22**, 35 (1987).

[9] F. Jones, `http://www.triumf.ca/compserv/accsim.html`, *User's Guide to ACCSIM*, 1990.

[10] J. Wei, *Longitudinal Dynamics of the Non-Adiabatic Regime on Alternating-Gradient Synchrotrons*, Ph.D thesis (1990).

[11] A. Luccio, *Spin Tracking in RHIC (Code Spink)*, Proceedings of the Adriatico Research Conference, World Scientific, 1997, p.235.

[12] M. Blaskiewicz, PRSTAB, Volume 1, 044201, 1998.

[13] R. Brun et al., *An Object-Oriented Data Analysis Framework*, `http://root.cern.ch`

[14] J.Irwin et al., *Model-Independent Beam Dynamics Analysis*, Physical Review Letters, Vol 82(8), 1999.

[15] `http://plasma-gate.weizmann.ac.il/Grace/`